TEC-0094

RADIUS: Model-Based Optimization

Lynn H. Quam, Sr. Pascal V. Fua
Thomas M. Strat, Sr. Aaron J. Heller
Christopher I. Connolly

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025-3493

March 1998

Approved for public release; distribution is unlimited.

Prepared for:
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington, VA 22203-1714

Monitored by:
U.S. Army Corps of Engineers
Topographic Engineering Center
7701 Telegraph Road
Alexandria, VA 22315-3864

US Army Corps of Engineers
Topographic Engineering Center

T
E
C

19980611 105

DTIC QUALITY INSPECTED 6

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1998	3. REPORT TYPE AND DATES COVERED Third Annual Oct. 1994 - Sep. 1995

4. TITLE AND SUBTITLE RADIUS: Model-Based Optimization	5. FUNDING NUMBERS DACA76-92-C-0034
6. AUTHOR(S) Lynn H. Quam, Sr. Pascal V. Fua Thomas M. Strat, Sr. Aaron J. Heller Christopher I. Connolly	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) SRI International 333 Ravenswood Avenue Menlo Park, CA 94025-3493	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive, Arlington, VA 22203-1714 U.S. Army Topographic Engineering Center 7701 Telegraph Road, Alexandria, VA 22315-3864	19. SPONSORING / MONITORING AGENCY REPORT NUMBER TEC-0094
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The construction and use of 3-D models of military and industrial sites will allow revolutionary advances in the speed, confidence, and range of analytical techniques with which an Image Analyst (IA) develops and reports intelligence information. This SRI research project, in support of the RADIUS Program, seeks to increase the speed and accuracy with which site models can be constructed from current imagery by developing a new family of image understanding techniques, and by developing a novel way for an IA to employ them.

14. SUBJECT TERMS Computer Vision Aerial Image Analysis RADIUS Optimization Snakes	15. NUMBER OF PAGES 39
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED

Contents

List of Figures

List of Tables

Preface

This research is sponsored by the Defence Advanced Research Projects Agency (DARPA) and monitored by the U.S. Army topographic Engineering Center (TEC) under Contract DACA76-92-C-0034, titled “RADIUS: Model-Based Optimization.” The DARPA Program Manager is Dr. Thomas M. Strat, and the TEC Contracting Officer’s Representative is Ms. Lauretta Williams.

1 Introduction

The construction and use of 3–D models of military and industrial sites will allow revolutionary advances in the speed, confidence, and range of analytical techniques with which an Image Analyst (IA) develops and reports intelligence information. This SRI research project, in support of the Research and Development for Image Understanding Systems (RADIUS) Program, seeks to increase the speed and accuracy with which site models can be constructed from current imagery by developing a new family of image understanding (IU) techniques, and by developing a novel way for an IA to employ them.

This research has proceeded on two fronts, simultaneously:

- Extending the generality and power of Model-Based Optimization (MBO) algorithms
- Developing a context-based approach to image analyst control of IU algorithms.

First we present our overall research goals and then discuss the progress toward advancing the state-of-the-art in these two areas.

2 Research Goals

Model-Supported Exploitation (MSE) is the analysis of imagery (by human or computer) with the aid of 2–D and 3–D models of the scene [10, 13, 17]. There are two major scientific problems that must be solved for MSE to be a viable concept for use in an operational intelligence setting. A successful MSE system must have:

- An interface that enables the image analyst (IA) to easily specify what he wants the machine to do
- A set of algorithms that enables the machine to perform the tasks posed by the IA.

While the majority of research in IU has been concerned with fully automated algorithms for interpreting images, the Perception Group at SRI has made the design and implementation of semiautomated systems a major goal.

- The MBO technology has ben extrended to provide the capability of extracting many different object classes under a wide variety of imaging and scene conditions. Integrated into the RADIUS Common Development Environment (RCDE), this technology constitutes an operational suite of tools tailored to the needs of the site model builder.
- The development of a software architecture that automatically chooses IU algorithms and their parameters given modeling tasks posed by an image analyst. The centerpiece of the approach is a framework that reasons about the context of the given task to make these choices intelligently.

3 Model-Based Optimization (MBO)

MBO is a paradigm in which an objective function is used to express both geometric and photometric constraints on features of interest. A model of a feature (such as a road, building, or coastline) is extracted from an image by adjusting the model until a minimum value of the objective function is obtained. The optimization procedure yields a description that simultaneously satisfies (or nearly satisfies) all constraints, and, as a result, is likely to be a good model of the feature.

Implementation of an MBO algorithm requires the specification of four components.

Objective function: A mathematic function that expresses the preferred geometric and photometric properties to be exhibited by the feature.

Representation: The geometric primitives used to represent the feature, which thereby limits the class of features that can be modeled.

Optimization: The procedure to be employed for finding a configuration of the feature that locally minimizes the objective function.

Initial conditions: The configuration of the feature to be used as the starting point by the optimization procedure.

This research addresses all four of these areas in seeking to find instantiations of the MBO paradigm that provide effective means for extracting features of interest to the RADIUS Program.

In the last twelve months the MBO package has been extensively reorganized and extended accommodate more complex types of objects. Earlier, we could only deal with polygonal curves that could be modeled as a sequential list of vertices. However, many objects that are of importance to a photoanalyst and are supported by RCDE, such as road networks or 3–D extruded objects, do not fit this model. Their topology is that of a network, and, to describe them completely, one must supply not only the list of their vertices, but also a list of "edges" that defines the connectivity of those vertices. In addition, with some of these complex objects one can define "faces," that is, circular lists of vertices that must be constrained to remain planar.

Therefore new breeds of snake were introduced—we refer to these deformable models as "generalized snakes" [12]—and can now accommodate the full taxonomy of snakes described in Table 1. The columns represent different types of snakes and the rows indicate different kinds of constraints that can be brought to bear. The table entries are examples of objects that can be modeled using these combinations. These generalized snakes are described in detail in Appendix A.

The effectiveness of the package was evaluated by instrumenting the code to record the amount of user intervention. We have found that using this package to model high-resolution roads leads to a fivefold reduction in effort as measured by the number of mouse clicks or mouse travel-time. These results are documented in Appendix B.

Table 1: Snake taxonomy. The columns represent different types of snakes and the rows different kinds of constraints that can be brought to bear. The table entries are examples of objects that can be modeled using these combinations.

Constraints/Type	Simple curve	Ribbon curve	Network
Smooth	Low res. roads, rivers	High res. roads	Road networks
Polygonal	Man-made structures	City streets	Street Networks
Planar	Planar structures	City streets	Street Networks
Rectilinear	Roof tops, parking lots	City streets	Buildings

Furthermore, a constrained-optimization scheme was developed that allows us to impose hard constraints on snakes. For example, we can ensure that two snakes are at a given distance from each other or that the altitude along the model of a river decreases monotonically [5].

4 Context-Based Architecture

Thirty years of research in IU have produced an enormous number of computer vision algorithms, many of which have demonstrated reliable performance at solving particular tasks in restricted domains. However, the development of computer vision systems that are reliable in more general circumstances has proved elusive [11]. It is in response to this situation that a framework is offered within which computer vision algorithms of specialized competence can be used and integrated with other such algorithms to produce a reliable vision system that operates effectively in a broader context than any of its individual component algorithms can. Recently, other authors have also stressed the use of context to aid image interpretation [1, 2, 3].

The site model construction and editing systems being developed within the RADIUS program comprise a large number of computer vision algorithms, each tailored to accomplish a particular task under a particular set of circumstances. The goals of these algorithms may overlap or even duplicate each other's goals, but the assumptions that they make and the data with which they operate will often differ. Automatically choosing suitable algorithms and parameter settings in order to solve particular IU tasks is important to minimize the understanding of IU technology that is required of a user who endeavors to use the algorithms. Within the RADIUS Program this concern is paramount because of the need to put advanced IU technology in the hands of casual users, specifically the image analysts.

The strategy for integrating such a collection of computer vision algorithms is based on prior work in context-based vision at SRI [18, 17, 19]. We represent explicitly the assumptions made by each algorithm, and use the context of the present task to select the most appropriate algorithms for solving that task. By doing so, we seek to avoid the source of many failures of computer vision techniques—the employment of an algorithm outside the bounds of its intended domain of competence. A prototype of a context-based architecture (CBA) has been implemented within RCDE which offers a design methodology with broad applicability. For example, CBA is an attractive framework for organizing a site model construction system using many algorithms that extract different features under different circumstances.

A prototype system has been implemented in the RCDE, known as the Hierarchical Update and Build (HUB) System. Logic programming was chosen to implement the context-based architecture because it provides a natural declarative language for expressing the constraints; unification provides a powerful mechanism for matching the contextual constraints (as encoded in the rules) to the current context; and the logical backchaining of a rule-based system provides the ability to search the rule base for algorithms that are applicable. Appendix C, provides a draft specification for integration of algorithms into the HUB that we will expand on and refine during the next phase of the project.

It is clear that the performance of an IU system employing the context-based architecture could also be attained by integrating the same computer vision algorithms via more traditional methods. However, the explicit representation of contextual constraints affords a number of additional benefits that would be lost to a purely functional integration. These benefits include:

Task specification: The context-based architecture allows the user to specify the task to be accomplished, leaving the selection of specific algorithms to be decided by the system. For example, the user can state that he would like the system to construct a 3-D model of a building, and the system would decide which of several building extraction algorithms would be most appropriate given the currently available imagery and auxiliary data. The user can make effective use of the IU system while possessing little knowledge of the capabilities and limitations of the individual computer vision algorithms.

Choosing parameters: The context rules are used to establish algorithm parameter settings, in the same way that they delimit the range of applicability. While computer vision algorithms can often compute their parameter settings from data at run time, the context rules provide a uniform means for all algorithms to specify how their parameters are to be determined.

Incremental integration: When building large systems in an evolutionary fashion, it can be difficult to add new capabilities without jeopardizing the integrity of existing capabilities. The modular decomposition of the context rules allows the developer to integrate a new algorithm by adding a packet of rules governing its use, without modifying existing code.

Choosing imagery: It is sometimes important to identify the imagery that is most likely to allow an algorithm to yield a desired result, rather than to choose an algorithm to run on a preselected image. The context rules already encode the information necessary to make this determination—they can be used to answer this question by fixing the algorithm and allowing the image to be a variable in the query. In fact, the context-rule base can be used to answer both questions simultaneously, finding the best combination of algorithms and images to satisfy a given task.

5 Conclusion

The RADIUS Program can benefit greatly from the use of more highly automated means for constructing and updating 3-D site models.

Research on model-based optimization has led to the development of a number of tools that increase the degree of automation and precision that is possible. These have been implemented in the RCDE and are being incorporated in the RADIUS Testbed. Further use of these tools within the Testbed will undoubtedly lead to new ideas and additional improvements in the site-model construction process.

Work on the context-based vision paradigm has led to the development of an architecture for integrating independently developed IU algorithms within a single system. The architecture provides the additional benefit of allowing the photoanalyst to specify a desired result, rather than a specific procedure to be followed, in carrying out a site model construction task.

Together, these developments, which were beyond the state-of-the-art at the outset of this project three years ago, have dramatically increased the feasibility of constructing and maintaining 3–D site models for use within the RADIUS Testbed.

A Cartographic Applications of Model-Based Optimization

Author: P. Fua
Published in the proceedings of the IU Workshop, February 1996.

1 Introduction

Model-Based Optimization (MBO) is a paradigm in which an objective function is used to express both geometric and photometric constraints on features of interest. A parametric model of a feature (such as a road, a building, or coastline) is extracted from one or more images by adjusting the model's state variables until a minimum value of the objective function is obtained. The optimization procedure yields a description that simultaneously satisfies (or nearly satisfies) all constraints, and, as a result, is likely to be a good model of the feature.

The deformable models we use here are extensions of traditional snakes [20, 12, 6]. They are polygonal curves or facetized surfaces to which is associated an objective function that combines an "image term" that measures the fit to the image data and a regularization term that enforces geometric constraints.

We model linear features as polygonal curves that may either be described as sequential list of vertices, or, for more complex objects, such as a road network or a 3-D extruded object, we must describe the network topology. In the latter case, to describe the object completely, one must supply not only the list of their vertices but also a list of "edges" that defines the connectivity of those vertices. In addition, with some of these complex objects, one can define "faces," that is, circular lists of vertices that must be constrained to remain planar.

Our ultimate goal is to accommodate the full taxonomy of snakes described in Table 2. The columns represent different type of snakes and the rows represent different kinds of constraints that can be brought to bear. The table entries are examples of objects that can be modeled using these combinations. The algorithms described below are implemented within the RADIUS Common Development Environment (RCDE) [14].

2 Polygonal Snakes

A simple polygonal snake, $\mathcal{C}$, can be modeled as a sequential list of vertices, that is, in two dimensions, a list of 2-D vertices $\mathcal{S}_2$ of the form

$$\mathcal{S}_2 = \{(x_i \; y_i), \; i = 1, \ldots, n\} \quad , \tag{1}$$

Table 2: Snake taxonomy. The columns represent different types of snakes and the rows represent different kinds of constraints that can be brought to bear. The table entries are examples of objects that can be modeled using these combinations.

Constraints/Type	Simple curve	Ribbon curve	Network
Smooth	Low res. roads, rivers.	High res. roads.	Road network.
Polygonal	Man-made structures.	City streets	Street Networks.
Planar	Planar structures.	City streets	Street Networks.
Rectilinear	Roof tops, parking lots.	City streets	Buildings.

and, in three dimensions, a list of 3–D vertices $\mathcal{S}_3$ of the form

$$\mathcal{S}_3 = \{(x_i \; y_i \; z_i), \; i = 1, \ldots, n\} \; . \tag{2}$$

In the two dimensional case, the "image energy" of these curves—the term we try to minimize when the optimization is performed, is taken to be

$$\mathcal{E}_I(\mathcal{C}) = -\frac{1}{|\mathcal{C}|} \int_0^{|\mathcal{C}|} |\nabla \mathcal{I}(\mathbf{f}(s))| \, ds, \tag{3}$$

where I represents the image gray levels, s is the arc length of $\mathcal{C}$, $\mathbf{f}(s)$ is a vector function mapping the arc length s to points (x, y) in the image, and $|\mathcal{C}|$ is the length of $\mathcal{C}$. In practice, $\mathcal{E}_I(\mathcal{C})$ is computed by integrating the gradient values $|\nabla \mathcal{I}(\mathbf{f}(s))|$ in precomputed gradient images along the line segments that connect the polygonal vertices.[1] We therefore rewrite $\mathcal{E}_I$ as

$$\mathcal{E}_I = \sum_{1 \leq i < n} \frac{S((x_i, y_i), (x_{i+1}, y_{i+1}))}{\sum_{1 \leq i < n} L_{i,i+1}} \quad , \tag{4}$$

where $L_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$ is the length of the individual line segments, and $S((x_i, y_i), (x_j, y_j))$ is

$$-\int_0^1 |\nabla \mathcal{I}(x_i + \lambda(x_j - x_i), y_i + \lambda(y_j - y_i))| \, d\lambda \; ,$$

the sum of the gradient values along one segment. It is computed by sampling the segment at regular intervals.

In the 3–D case, $\mathcal{E}_I(\mathcal{C})$ is computed by projecting the curve into a number of images, computing the image energy of each projection and summing these energies. Formally,

[1]The gradient images are computed by gaussian smoothing the original image and taking the x and y derivatives to be finite differences of neighboring pixels.

given a set of N images and corresponding camera models, we write

$$\mathcal{E}_I = \sum_{1 \le k \le N} \mathcal{E}_I^k \ , \tag{5}$$
$$\mathcal{E}_I^k = \sum_{1 \le i < n} \frac{S(P^k(x_i, y_i, z_i), (P^k(x_{i+1}, y_{i+1}, z_{i+1})))}{\sum_{1 \le i < n} L_{i,i+1}^k} \ ,$$

where k denotes the image number, $P^k(x, y, z)$ the pair of coordinates of the projection of point (x, y, z) into image k and $L_{i,j}^k$ the length of the projection into image k of the segment i, j.

3 Smooth Snakes and Ribbons

These snakes are used to model smoothly curving features, such as roads or ridge-lines.

2–D curves. Following Kass *el al.* [12], we choose the vertices of such curves to be roughly equidistant and add to the image energy $\mathcal{E}_I$ a regularization term $\mathcal{E}_D$ of the form

$$\begin{aligned} \mathcal{E}_D(\mathcal{C}) &= \mu_1 \sum_i (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 \\ &+ \mu_2 \sum_i (2x_i - x_{i-1} - x_{i+1})^2 \\ &\quad +(2y_i - y_{i-1} - y_{i+1})^2 \end{aligned} \tag{6}$$

and define the "total energy" $\mathcal{E}_T$ as

$$\mathcal{E}_T(\mathcal{C}) = \mathcal{E}_D(\mathcal{C}) + \mathcal{E}_I(\mathcal{C}) \tag{7}$$

The first term of $\mathcal{E}_D$ approximates the curve's tension and the second term approximates the sum of the square of the curvatures, assuming that the vertices are roughly equidistant. In addition, when starting with regularly spaced vertices, this second term tends to maintain that regularity. To perform the optimization we could use the steepest or conjugate gradient, however it would be slow for curves with large numbers of vertices. Instead, it has proven much more effective to embed the curve in a viscous medium and solve the equation of the dynamics

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial S} + \alpha \frac{dS}{dt} &= 0, \\ \text{with } \frac{\partial \mathcal{E}}{\partial S} &= \frac{\partial \mathcal{E}_D}{\partial S} + \frac{\partial \mathcal{E}_I}{\partial S}, \end{aligned} \tag{8}$$

where $\mathcal{E}$ is the energy of Equation 7, α the viscosity of the medium, and S the state vector that defines the current position of the curve. Since the deformation energy $\mathcal{E}_D$ in

Equation 6 is quadratic, its derivative with respect to S is linear, therefore, Equation 8 can be rewritten as

$$\begin{aligned} K_S S_t + \alpha(S_t - S_{t-1}) &= -\left.\frac{\partial \mathcal{E}}{\partial S}\right|_{S_{t-1}} \\ \Rightarrow (K_S + \alpha I) S_t &= \alpha S_{t-1} - \left.\frac{\partial \mathcal{E}}{\partial S}\right|_{S_{t-1}} \end{aligned} \tag{9}$$

where

$$\frac{\partial \mathcal{E}_D}{\partial S} = K_S S,$$

and K_S is a sparse matrix. Note that the derivatives of $\mathcal{E}_D$ with respect to x and y are decoupled so that we can rewrite Equation 9 as a set of two differential equations in the two spatial coordinates

$$\begin{aligned} (K + \alpha I) X_t &= \alpha X_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial X}\right|_{X_{t-1}} \\ (K + \alpha I) Y_t &= \alpha Y_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial Y}\right|_{Y_{t-1}} \end{aligned}$$

where K is a pentadiagonal matrix, and X and Y are the vectors of the x and y vertex coordinates. Because K is pentadiagonal, the solution to this set of equations can be computed efficiently in $O(n)$ time using LU decomposition and backsubstitution. Note that the LU decomposition need be recomputed only when α changes.

In practice α is computed in the following manner. We start with an initial step size Δ_p, expressed in pixels, and use the following formula to compute the viscosity:

$$\alpha = \frac{\sqrt{2n}}{\Delta_p} \left| \frac{\partial \mathcal{E}}{\partial S} \right|, \tag{10}$$

where n is the number of vertices. This ensures that the initial displacement of each vertex is on the average of magnitude Δ_p. Because of the non linear term, we must verify that the energy has decreased from one iteration to the next. If, instead, the energy has increased, the curve is reset to its previous position, the step size is decreased, and the viscosity recomputed accordingly. This is repeated until the step size becomes less than some threshold value. In most cases, because of the presence of the linear term that propagates constraints along the whole curve in one iteration, it takes only a small number of iterations to optimize the initial curve.

3–D Curves. To extend the smooth snakes to three dimensions, we add one term in z to the deformation energy of Equation 6 and $\mathcal{E}_D$ becomes

$$\mathcal{E}_D(\mathcal{C}) = \mu_1 \sum_i (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 \tag{11}$$

$$+(z_i - z_{i-1})^2 + \mu_2 \sum_i (2x_i - x_{i-1} - x_{i+1})^2$$
$$+(2y_i - y_{i-1} - y_{i+1})^2 + (2z_i - z_{i-1} - z_{i+1})^2$$

Since the derivatives of $\mathcal{E}_D$ with respect to x, y, and z are still decoupled, we can rewrite Equation 9 as a set of three differential equations in the three spatial coordinates:

$$\begin{aligned}(K+\alpha I)X_t &= \alpha X_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial X}\right|_{X_{t-1}}\\ (K+\alpha I)Y_t &= \alpha Y_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial Y}\right|_{Y_{t-1}}\\ (K+\alpha I)Z_t &= \alpha Z_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial Z}\right|_{Z_{t-1}}\end{aligned}$$

where X,Y, and Z are the vectors of the x,y, and z vertex coordinates.

The only major difference with the 2–D case is the use of the images' camera models. In practice, $\mathcal{E}_I(\mathcal{C})$ is computed by summing gradient values along the line segments linking the vertices' projections. These projections, and their derivatives, are computed from the state vector S using the camera models. Similarly, to compute the viscosity, we use the camera models to translate the average initial step Δ_p, a number of pixels, into a step Δ_w expressed in world units and use the latter in Equation 10.

Ribbons 2–D snakes can also be extended to describe ribbon-like objects such as roads in aerial images. A ribbon snake is implemented as a polygonal curve forming the center of the road. Associated with each vertex i of this curve is a width w_i that defines the two curves that are the candidate road boundaries. The state vector S becomes the vector $S = \{(x_i\ y_i\ w_i)\},\ i = 1,\dots,n\}$ and the average edge strength the sum of the edge strengths along the two boundary curves. Since the width of roads tends to vary gradually, we add an additional energy term of the form

$$\begin{aligned}\mathcal{E}_W(\mathcal{C}) &= \sum_i (w_i - w_{i-1})^2 \qquad (12)\\ \Rightarrow \frac{\partial \mathcal{E}_W}{\partial W} &= LW,\end{aligned}$$

where W is the vector of the vertices' widths and L a tridiagonal matrix. The total energy can then be written as

$$\mathcal{E}(\mathcal{C}) = \lambda_D \mathcal{E}_D(\mathcal{C}) + \lambda_W \mathcal{E}_W(\mathcal{C}) + \lambda_G \mathcal{E}_I(\mathcal{C})$$

where λ_D and λ_W wheigh the contributions of the two geometric terms. At each iteration the system must solve the three differential equations:

$$(K+\alpha I)X_t = \alpha X_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial X}\right|_{X_{t-1}}$$

$$
\begin{aligned}
(K+\alpha I)Y_t &= \alpha Y_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial Y}\right|_{Y_{t-1}} \\
(K+\alpha I)W_t &= \alpha W_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial W}\right|_{W_{t-1}}
\end{aligned}
$$

2–D ribbons can be turned into 3–D ones in exactly the same way 2–D snakes are turned into 3–D ones. The state vector S becomes the vector $S = \{(x_i\ y_i\ z_i\ w_i)\},\ i = 1,\ldots,n\}$ and at each iteration the system must solve four differential equations, one for each coordinate.

4 Network Snakes

The 2–D and 3–D "network snakes" are a direct extension of the polygonal snakes of Section 2.

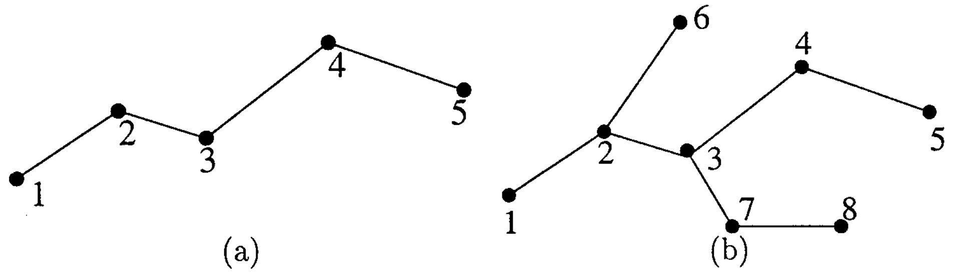

Figure 1: Snake Topology. (a) A simple polygonal curve is described by a sequential list of vertices v_i— $v_{1\leq 5}$ here. (b) A network is decribed by a list of vertices v_i— $v_{1\leq 8}$ here , and a list of edges—((1 2) (2 3) (3 4) (4 5) (2 6) (3 7) (7 8)) here.

In the 2–D, the extension is straightforward. A network snake is now defined by a list of n vertices $\mathcal{S}_\in$ as before and list of edges $\mathcal{A} = \{(i,j)$ where $1 \leq i \leq n$ and $1 \leq j \leq n\}$. Figure 1 depicts such a network snake. $\mathcal{E}_I(\mathcal{C})$ is computed as

$$\mathcal{E}_I(\mathcal{C}) = \sum_{(i,j)\in\mathcal{A}} S((x_i,y_i)(x_j,y_j)) / \sum_{(i,j)\in\mathcal{A}} L^k_{i,j} \ , \qquad (13)$$

where S and L are the functions defined in Equation 4. It is optimized using either steepest gradient descent or conjugate gradient. In Figure 2, we show an example of such a network. When constraints, such as planarity or rectilinearity, are imposed on the network, constrained optimization can also be used [9].

In the 3–D case, one must take into account the fact that not all the network's edges are visible in all views. As a result, one must also provide a list of visible edges for

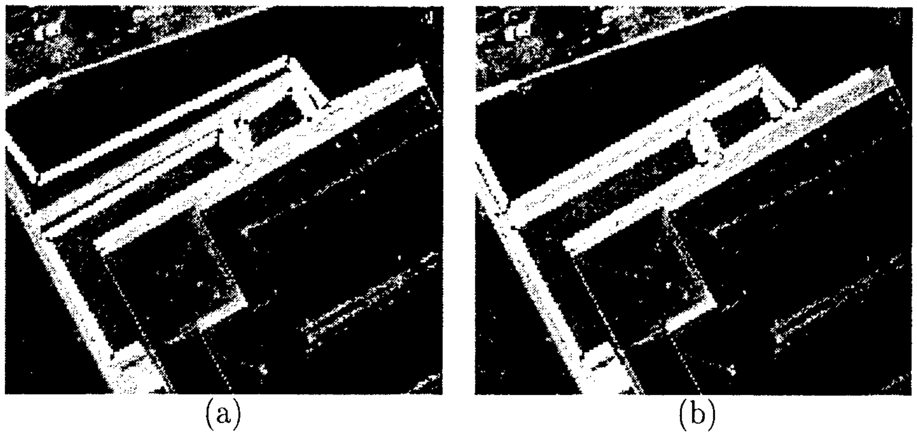

Figure 2: Optimizing a 2–D polygonal network. (a) Initialization (b) Network after optimization.

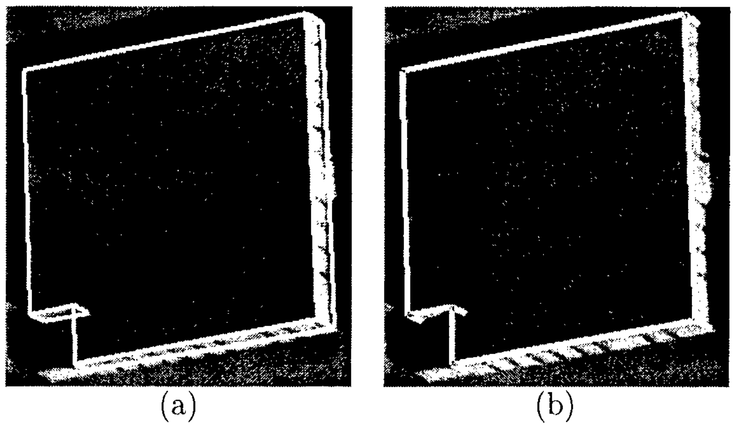

Figure 3: Edge Visibility. (a) An RCDE "extruded-object." Only the visible faces, that is those whose surface normal is oriented towards the viewer, are drawn. Note that this heuristic does not account for non-convexity. As a result the faces in the lower left corner of the image are improperly drawn. (b) The network snake that is generated to optimize the extruded-object. It includes roof-edges and vertical wall-edges. The edges at the back of the building are not drawn—and not used during the computations involving these views—because they belong to hidden faces. The edges at the base of the building are treated as invisible because their appearance is unreliable in typical imagery.

each projection of the snake into the set of images images. This list is computed by using the face-visibility methods embedded in RCDE: we assume that only edges that belong to visible faces are visible. This is effective for convex objects but may fail for concave ones. Figure 3 illustrates the strengths and weaknesses of this approach. A better way to compute visibility would be to use the Z-buffering capabilities of SGI machines; unfortunately this is impractical for the time being because the graphics libraries supplied by SGI cannot currently be loaded into RCDE, due to limitations of the Lucid Common Lisp compiler.[2]

Formally, given a set of N images, we define a visibility list $\mathcal{A}^k_{1\leq k\leq N}$ for each image and we rewrite the image energy of Equation 6 as

$$\begin{aligned} \mathcal{E}_I &= \sum_{1\leq k\leq N} \mathcal{E}_I^k \ , \\ \mathcal{E}_I^k &= \sum_{(ij)\in\mathcal{A}^k} \frac{S(P^k(x_i,y_i,z_i),(P^k(x_{i+1},y_{i+1},z_{i+1})))}{\sum_{1\leq i<n} L^k_{i,i+1}} \ . \end{aligned} \tag{14}$$

The optimization typically is a three-step process as illustrated in Figure 4:

1. We optimize a snake using a single image. Since a single view underconstrains the three degrees of freedom of the individual vertices, we fix one of them for each vertex. The z value is fixed and only the x and y coordinates are allowed to change. As shown in Figure 4(b), at the end of this first step, the network's projections match image features in the view that was used but not necessarily in any other view because the fixed z values usually are erroneous.
2. The height of the vertices is estimated by assuming that the network is horizontal and searching through a range of z values, calculating the x and y values for each vertex so that the projection of the network remains the same in the view used in the previous step and retaining the z value that yields the optimal value of $\mathcal{E}_I$, the image energy of Equation 15.
3. The 3–D positions of the network's vertices are further refined by optimizing $\mathcal{E}_I$ with respect to all three degrees of freedom of the vertices simultaneously.

As in the case of the 2–D networks, the optimization of Steps 1 and 3 can be performed using either steepest gradient descent, conjugate gradient or constrained optimization.

The number of degrees of freedom of generic 3–D networks can be reduced by forcing them to be planar. We do this either by defining a plane of equation

$$z = ax + by + c \tag{15}$$

and imposing that the vertices lie on such a plane, or imposing planar constraints on sets of four vertices. In both cases, we replace the n degrees of freedom necessary to specify the elevation of each vertex by the three degrees of freedom required to define the plane.

These 3–D networks can be further specialized to handle objects that are of particular interest in urban environments: trihedral corners found on building roofs and extruded objects that are used in RCDE to model building outlines.

[2]This limitation has been removed in version 5.0 of the Lucid Common Lisp compiler.

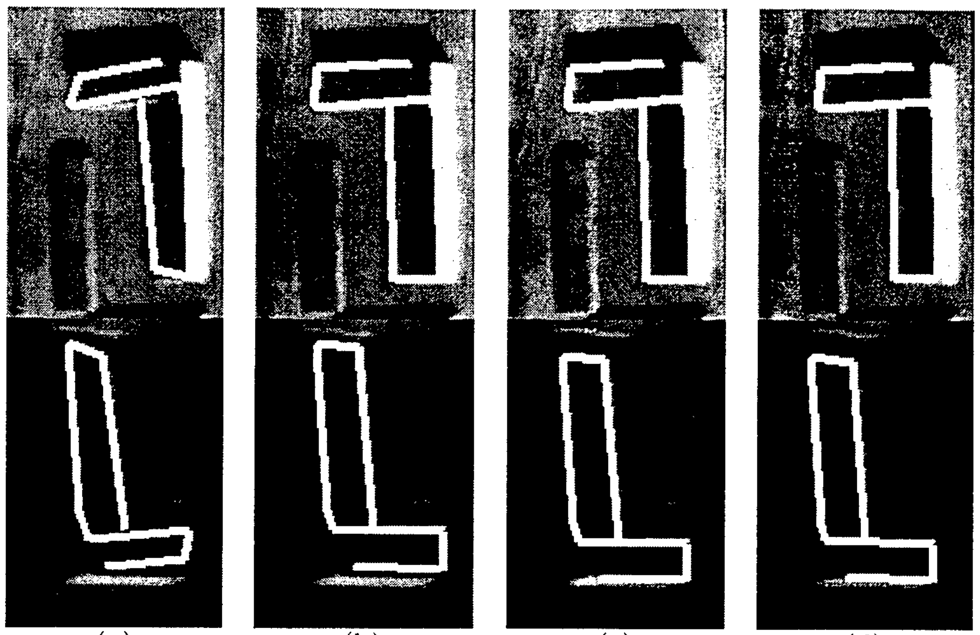

Figure 4: Three-stage optimization of a 3–D Network. (a) The object is hand-entered using RCDE. By default the vertex heights are that of the underlying terrain model. (b) The object is optimized using only the top view. The object matches the roof outline in the top view but not the lower one because the object is higher than the terrain. (c) The height of the vertices is computed approximately by searching a range of z values while maintaining the shape of the object's projection in the top view. (d) The network's 3–D shape is further refined by simultaneously optimizing the x, y and z values of the vertices' positions. Its projections then match image features in both views, guaranteeing that the 3–D shape of the underlying objects has been recovered.

Trihedral corners. They are modeled as networks with four vertices and three edges forming 90 degree angles with each other, as shown in Figure 5. We typically impose the additional constraint that one edge be vertical while the two other are horizontal. Under such constraints, the trihedral corner has only four degrees of freedom: three for the position of the vertex that is shared by all three edges and one for rotation about the vertical axis. When optimizing using only one image, fixing the altitude removes one additional degree of freedom. In both cases, the optimization is much more constrained than for generic 3–D networks, and, as a result, the convergence properties are substantially improved.

In Figure 6, we show the recovery of such a trihedral corner. Figure 7 shows additional corners recovered and superimposed on a manually-entered 3–D wireframe model of the corresponding buildings. Because the corners are fully 3–D objects, they can be

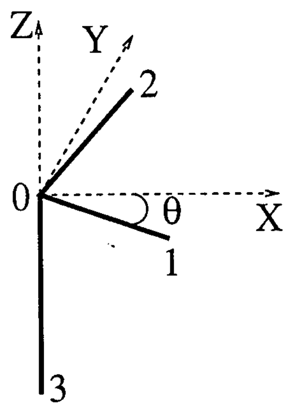

Figure 5: Topology of a trihedral corner. It has four vertices and three edges that all share one vertex. The edges form 90 degree angles. Two of them may be constrained to be horizontal. In this case the corner has only four degrees of freedom, three for the position of vertex 0, and 1 for the orientation of the horizontal edges.

viewed from different viewpoints in which they match the 3–D structure of the underlying objects.

Note that in order to accurately recover the corner's 3–D position, the camera models associated with the images must be fairly precise—which they are in the examples presented here. However, if the camera models were less accurate, we could still perform the single-view optimization in each image separately. We could then feed the results of optimizing several corners to a resection program and refine the camera models.

Extruded objects. Extruded objects are typically used to model buildings such as those in Figure 7. For optimization purposes, we define extruded networks that are composed of a polygonal closed contour that corresponds to the roof outline and of vertical edges that correspond to the intersections of the vertical walls as shown in Figure 8. As discussed above (see Figure 3), for each view, k, in which the extruded object is visible, we define a list $\mathcal{A}^k$ of edges that are visible and use only those to compute the image energy $\mathcal{E}_I$.

During the optimization, the "wall" edges are constrained to remain vertical. We can also constrain the "roof-outline" to be planar and the "roof-edges" to form 90-degree angles. As in the case of 3–D corners, these constraints greatly reduce the number of degrees of freedom and allow for better convergence properties.

Figure 9 illustrates the recovery of a building using all the constraints described above. For comparison's sake, in Figure 10, we show the result of the optimization using the same starting point without imposing the rectilinearity constraint.

Figure 11 shows several buildings modeled by roughly entering their outlines within

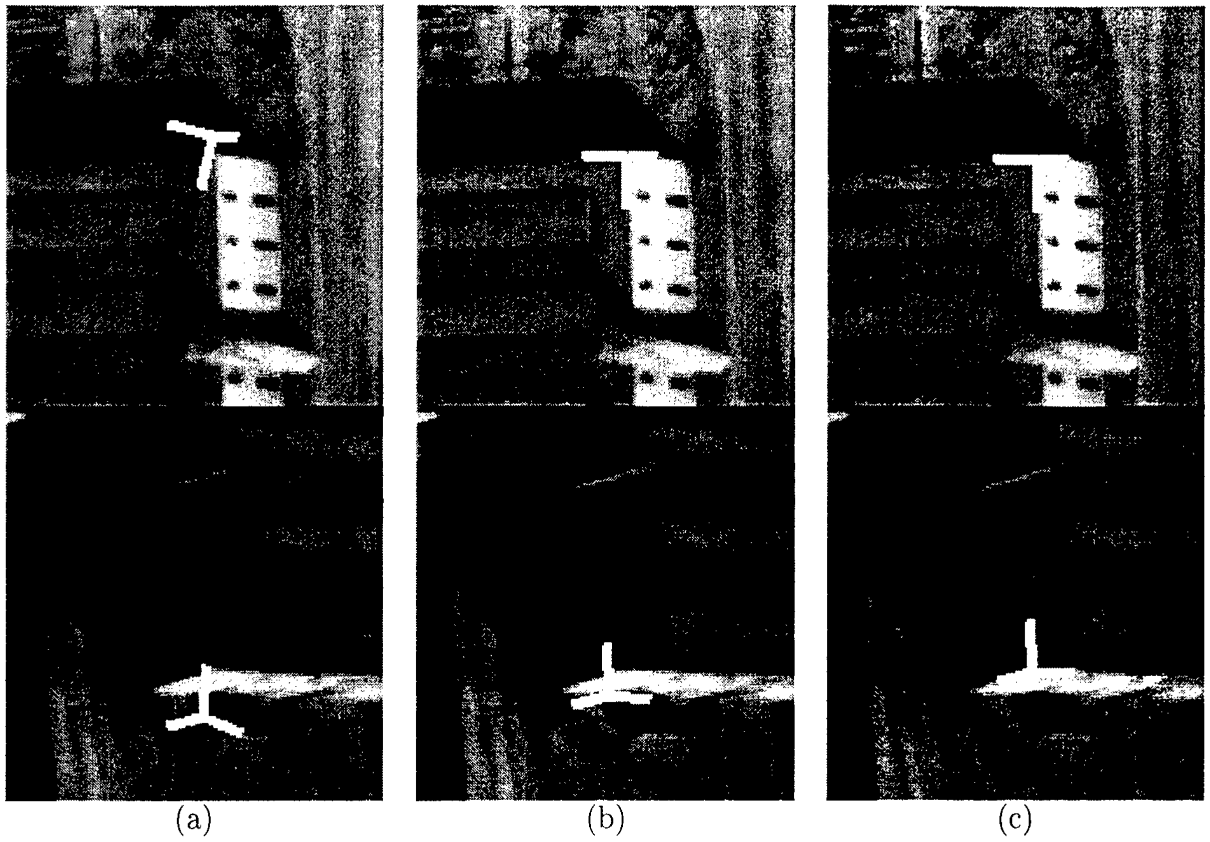

Figure 6: Optimization of a trihedral corner. (a) Initial position. (b) After optimization using only the top view, the corner's projection matches the image features in the top view only. (c) After optimization using both views, the corner's projections match the image features in both views.

RCDE and optimizing the shapes in three views simultaneously using our extruded snakes. The use of the snakes has allowed us to perform this task much faster than we would have if we had to precisely delineate all five buildings by hand.

5 Conclusion

We have presented object modeling techniques for 2–D and 3–D linear features that rely on parametric models that are extensions of traditional snakes. Using a variety of real imagery, we have demonstrated that the resulting methods allow powerful and flexible reconstruction.

Under a different contract, we have incorporated our MBO approach to surface reconstruction method [7, 8] into RCDE: surfaces are represented as hexagonally connected meshes of 3–D vertices that can be moved to minimize a stereo score. As a result, MBO

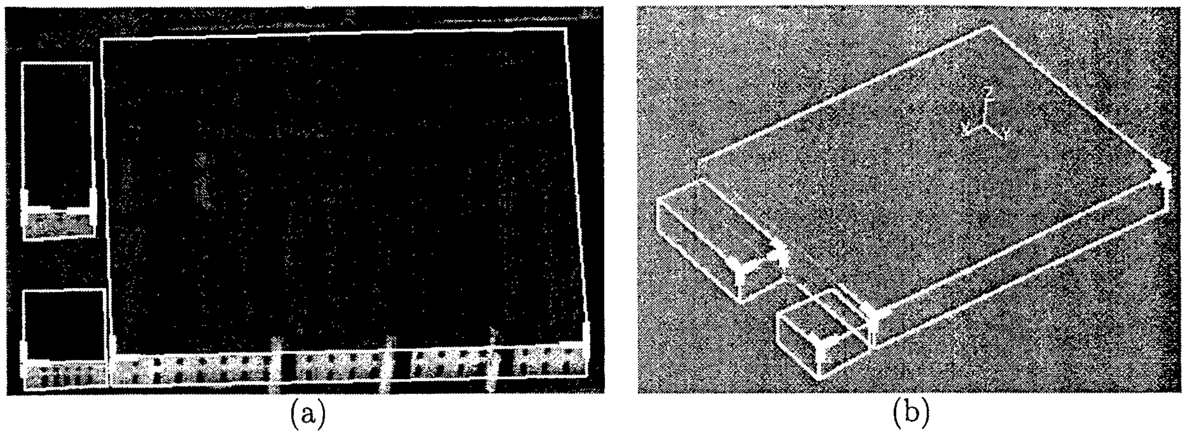

Figure 7: Recovering building corners. (a) Several building corners superimposed on manually entered 3–D wireframe models of the buildings. (b) The same corners and wireframes seen from a different viewpoint. Note that the recovered corners are 3–D objects that match the underlying objects.

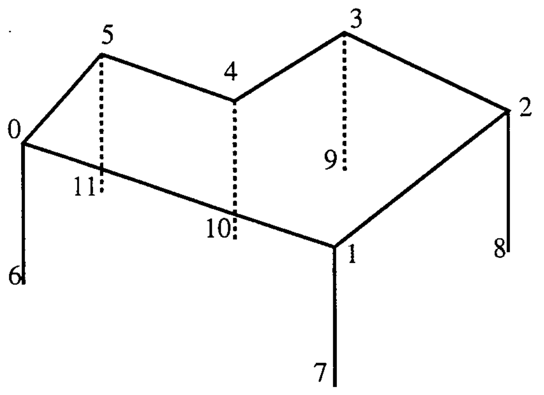

Figure 8: Topology of an extruded object. It has a polygonal outline that corresponds to the roof outline and vertical edges that correspond to the intersections of the vertical walls. In this example the complete edge-list is ((0 1) (1 2) (2 3) (3 4) (4 5) (5 0) (0 6) (1 7) (2 8) (3 9) (4 10) (5 11)). Note, however, that, because of occlusions, the list of visible edges for the particular projection shown here would be the sublist ((0 1) (1 2) (2 3) (3 4) (4 5) (5 0) (0 6) (1 7) (2 8)). The visible edges are shown as solid lines, and the hidden ones as dashed lines.

can now be used to simultaneously optimize several models, such as terrain, roads, rivers, and buildings, while enforcing consistency constraints between them [9], and produce realistic site-models of sites with both rugged terrain and man-made or natural features.

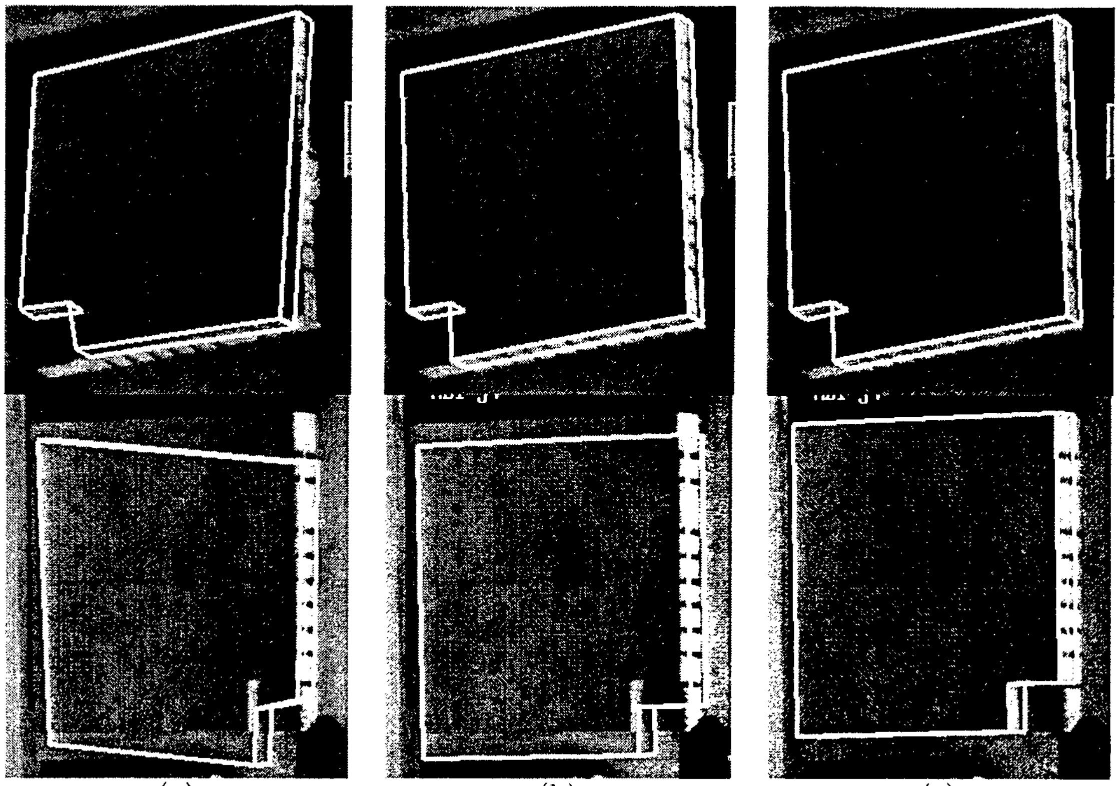
(a) (b) (c)

Figure 9: Optimization of an extruded object. (a) Initial position. (b) The edges are assumed to form 90 degrees angles. After optimization using only the top view, the object's projection matches the image features in the top view only. (c) After optimization using both views, the object's projections match the image features in both views.

We believe that this capability will prove indispensable to automating the generation of complex object databases from imagery, such as the ones required for realistic simulations or intelligence analysis.

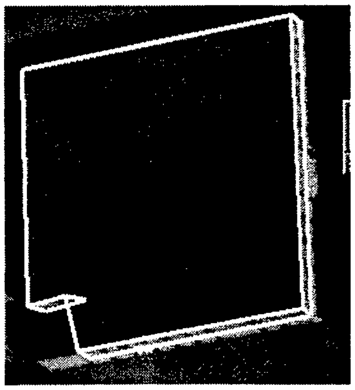

Figure 10: Extruded object of Figure 9 optimized without imposing the constraint that the roof-edges form 90-degree angles. The initial position was the one shown in Figure 9(a). Note that the corner in the lower left is not properly recovered.

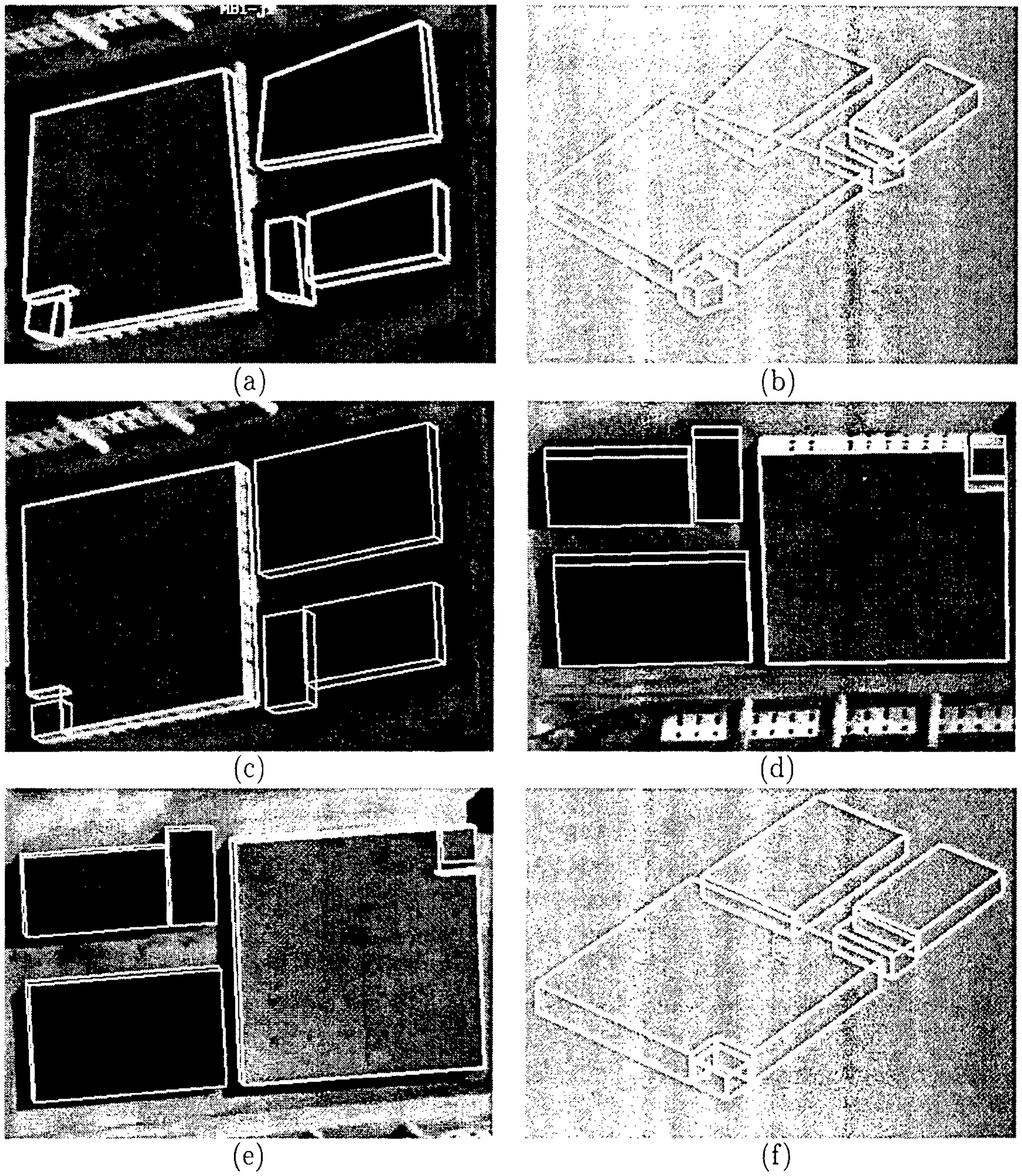

Figure 11: Buildings modeled by entering rough models within RCDE and optimizing them using the extruded-snakes. (a) Rough initial sketches overlaid on one of the images. (b) A view from a different perspective. (c,d,e) Final building outlines overlaid on the three images we used to perform the 3–D optimization. (f) A view of the buildings from the perspective of (b).

B Effectiveness Model-Based Optimization Algorithms

Authors: P. Fua, L. Quam and A. Heller
Published in the proceedings of the IU Workshop, February 1996.

1 Introduction

We have chosen to measure the amount of effort expanded by the human analyst by the number of mouse-clicks and the amount of mouse-travel required to achieve a desired answer. We feel that this is a better measure than, for example, actual computation times because it truly reflects the amount of human interaction and does not depend on the speed of the computer being used.

First, we briefly describe the code instrumentation that was required to perform the experiments and then report our results.

2 Instrumenting RCDE

Code was developed and installed that captures low level information from the RCDE user interface about individual actions taken by the analyst. Every mouse motion associated with making adjustments to object parameters, and every mouse click is captured into an event history. Below is a list of the information being recorded:

- Object Adjustment Events:
  - Object ID
  - Event start time
  - Adjustment type: (for example: vertex-xy, vertex-z, vertex-width, ...)
  - Two-dimensional world ID
  - Zoom level
  - Sequence of time deltas and mouse-position deltas of the form: (delta-t dx dy)
- Mouse Click Events:
  - Object ID
  - Event start time
  - Event ID (for example: zoom-in, zoom-out, recenter, drop-z)
  - Two-dimensional world ID
  - Zoom level

- Mouse 2–D world position

This event history was summarized by a small number of meaningful numbers. Among them were:

- Number of mouse clicks
- Number of mouse moves
- Total distance mouse moved during adjustments
- Total time in adjustment events
- Total time in fine adjustments

A metric to estimate the precision of an extracted road by comparing the centerline of the extracted road to the ground truth centerline was also implemented. For each vertex of the extracted road, the distance to the nearest centerline point of the ground truth centerline is computed. The data is reduced to the following two numbers:

- Mean vertex to ground truth distance
- Maximum vertex to ground truth distance

These seven numbers appear in the figures of the following section.

3 Experimental Results

We used images, such as the $7000 by 7000$ pixel image shown in Figure 12 to perform our experiments and chose a set of road segments to be modeled as accurately as possible. We compared four approaches to road delineation:

Hand Hand-tracing using the RCDE interface using neither snakes nor road tracker.

Tracker Using SRI's road tracker [16] to provide the initial sketch in the full resolution image, then refining it using a ribbon snake [4].

Snake1 Sketching the road using the full resolution image and refining it using a ribbon snake.

Snake2 Sketching the road using a half-resolution version of the image, refining it using a ribbon snake first at half-resolution, then at full resolution.

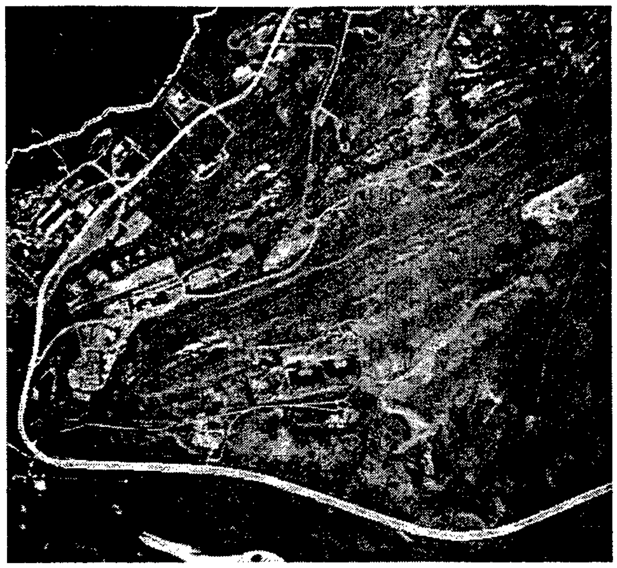

Figure 12: An image with two overlaid roads.

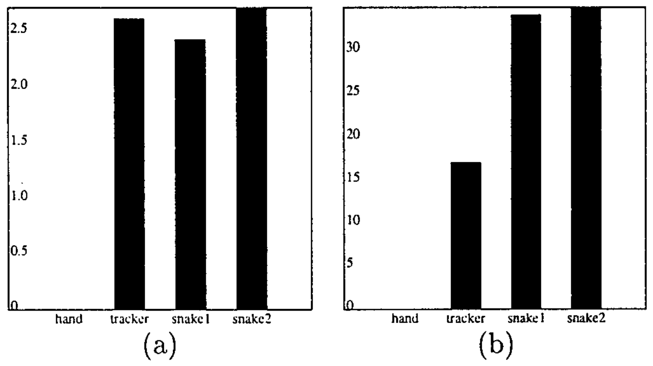

Figure 13: Distance to hand-entered roads. (a) Average distance difference. (b) Maximum distance difference. Because the hand-enetered results are taken to be the reference, the corresponding distances are zero.

In all cases we used the system's default parameter settings and allowed the user to manually refine the automatically generated results to produce satisfactory delineations. The bar graphs that appear in Figures 13 and 14 are labeled "hand," "tracker," "snake1," "snake2."

We used the hand-traced versions of the roads as the references and the metric discussed above to evaluate the quality of the delineations produced by the three semiautomated approaches. As shown in Figure 13, the results are virtually indistinguishable in terms of average distance whereas the "tracker" approach does better in terms of maximum distance.

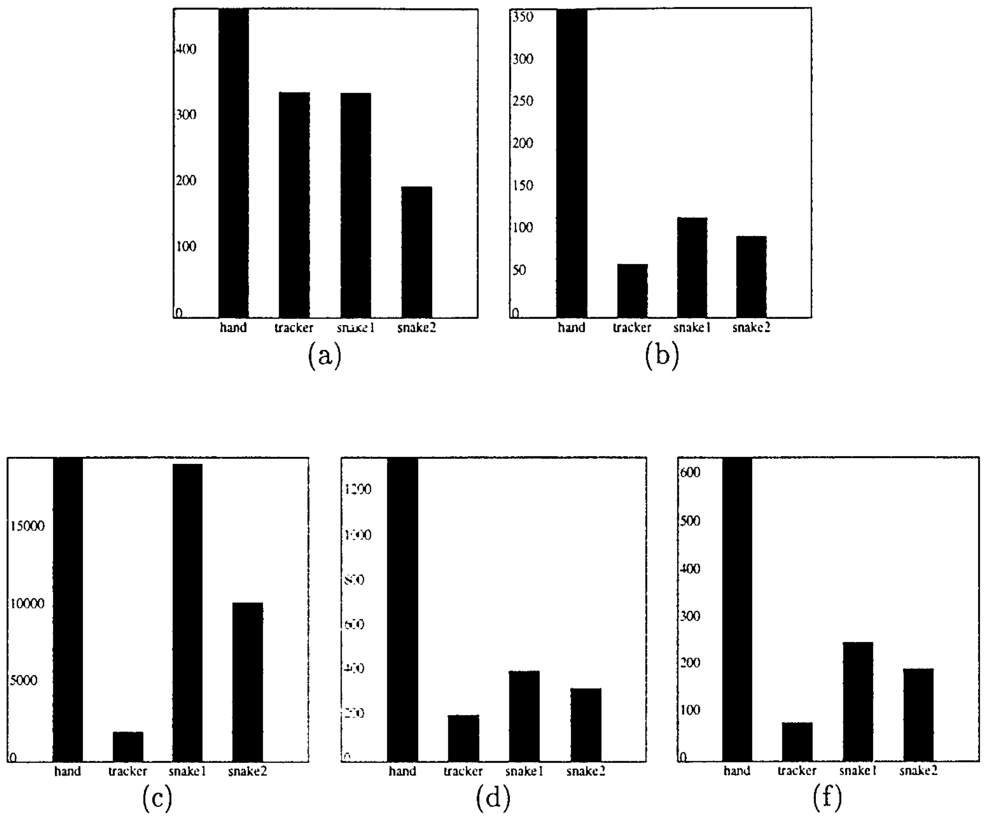

Figure 14: Amount of effort. (a) Number of object clicks. (b) Number of mouse moves. (c) Total mouse distance. (d) Total mouse move time. (e) Total near mouse move time.

Figure 14 depicts the amount of effort required by each approach, as measured by the number of mouse clicks and the amount of mouse travel. The tracker approach appears to be very effective and yields at least a sixfold improvement on all counts except the total number of mouse clicks. This is so because starting and stopping each operation—automated sketching and snake refinement—requires several clicks. This number could be drastically reduced by defining common-lisp methods that sequentially perform all these operations with a single mouse-click.

The "snake2" approach is almost as effective, however it requires more effort to provide the initial sketch. This problem could be alleviated by using ziplock snakes [15] instead of traditional snakes.

For all three semiautomed methods, however, a large portion of the human interaction goes into specifying the width of the road because current tools have no way of computing it. Therefore, methods to compute the width of a road given only its centerline would be extremely valuable and should be the object of future research.

In short, by further improving the interface and developing a width-computing algorithm, we should be able to turn the current sixfold reduction of effort into a ten to hundred-fold one.

C Integrating IU Algorithms into the RADIUS HUB

Author: C. Connolly

1 Introduction

This is a draft specification for integration of algorithms into the Hierarchical Update and Build (HUB) system. The HUB is a framework that allows the semi-automatic selection of *appropriate* algorithms for certain task and site combinations. In effect, this will help prevent algorithms from failing under circumstances for which they weren't designed. In addition, the HUB can select which images are appropriate for a given algorithm.

A prototypical example used in this document is road detection: the SRI road tracker works well on clearly defined roads at relatively high resolutions. For low-resolution images, where roads appear as thin curves, it is more appropriate to rely on ziplock snakes. Hence, the first example of HUB integration and usage will employ these two algorithms, allowing us to explore some of the issues that arise.

The HUB consists of a modified Prolog engine that contains rules and facts pertaining to images, sites, algorithms, and user preferences. The Prolog engine has been modified to allow "hooks" to be executed at each inference. This allows the HUB to keep track of successes and failures, and is intended to provide the user with useful feedback on the inference process (e.g., if no algorithm strictly satisfies the user's requirements, are there any that come close, say by one predicate). Based on the task to be performed, the HUB can query its database to determine which algorithm is suitable for applying on a given image, or for that matter, which images are suitable for running a particular algorithm.

The rule database must be populated. This requires us to specify exactly what kind of information is needed to integrate algorithms into the HUB. Algorithm designers will be asked to fill out a form, perhaps using the World Wide Web, that specifies the rule packet for a particular algorithm.

2 Vocabulary

The first step in specifying HUB rules is to define a suitable vocabulary for rules. The vocabulary is used to describe images, sites, features, algorithms, and user preferences, to allow the HUB to select algorithms to employ in a particular task.

There are four main categories in the vocabulary:

- Algorithm

- Image
- Site
- Feature

Although the HUB itself uses Prolog, algorithm designers are not required to specify their rule packets in Prolog. For inclusion in the HUB, each algorithm should be accompanied by a set of basic declarations of the properties of the algorithm, followed by 1 or more lists of image and site conditions under which the algorithm will exhibit robust behavior.

2.1 Algorithm Vocabulary

The algorithm vocabulary is used for basic declarations of algorithm properties. The algorithm designer should supply as many of these as is practical:

- `Time` – estimate of running time (algorithm speed)
- `Memory` – estimate of memory usage (kilobytes)
- `Disk` – estimate of disk usage (kilobytes)
- `Task-type` – e.g., extract, refine, or level of required initialization
- `Interactivity` – e.g., manual, semi-automatic, automatic
- `Accuracy` – e.g., coarse, fine

2.2 Image Vocabulary

At present, the image vocabulary contains the following image properties:

- `Dimensions` – image dimensions (numbers)
- `Time-of-day` – time-of-day and date
- `Imaging-geometry` – azimuth, elevation (degrees)[3]
- `Registration-error` – (number)
- `GSD` – ground sample distance – pixel width in meters at ground level (number)
- `Footprint` – ground area covered by the image (number)

[3]Intended for algorithms whose performance differs under nadir and off-nadir conditions.

- `Dynamic-range` – difference of maximum and minimum pixel values (number)
- `Cloud-cover` – percentage of cloud cover (number)
- `Insolation` – INcoming SOLar radiATION W/m^2 (number)
- `Spectrum` – portion of spectrum imaged (lo, hi, numbers)
- `Snow-cover` – percentage of snow cover in the image (number)
- `Albedo` – percentage of light reflected (number)

2.3 Site Vocabulary

There are a number of site-specific properties that form their own vocabulary:

- `Facility` – type of facility
- `Layout` – site layout (e.g., regular, random)
- `Direction` – dominant direction (degrees from north)
- `Vegetation` – percentage of site area
- `Buildings` – percentage of site area
- `Terrain` – flat, hilly, rugged, mountainous
- `Terrain-quality`:
  - fake-horizontal
  - fake-planar
  - DTED-1
  - DTED-2
  - hi-resolution
- `Georeferencing` – absolute, relative, none

2.4 Feature Vocabulary

This vocabulary is used to describe the kinds of features that can be extracted or refined by the algorithm:

- `Feature-type` – name of the feature type (e.g., road, building).

3 Rule Packets

The vocabulary described in the previous section forms the basis for defining a set of predicates that apply to images, sites, algorithms, and features. These predicates are then combined to form rules that dictate when an algorithm can be applied.

The algorithm developer has the responsibility of declaring which properties in the vocabulary apply to the algorithm. A sample rule packet has been written for the Road Tracker, which works well on relatively high resolution images:

```
algorithm: (smc::road-tracker1 image feature
                               &optional pane)

interactivity:          semiautomatic
accuracy:               coarse

arguments:
      image             image
      feature           road

requirements:
      task              extract
      feature-type      road
      gsd               < 1.5
```

Rule packets for an algorithm consist of simple declarations from the algorithm vocabulary (e.g., interactivity, accuracy) followed by an argument constraint list, followed by 1 or more requirements lists. Argument constraint lists declare the types and value constraints for arguments to the algorithm. Each requirements list describes prerequisites for applying the algorithm to a particular task.

Numerical properties can be represented as numbers (equalities), inequalities, or ranges, e.g.,

- 3.5
- > 2.3
- 1.1-7.4

Non-numerical properties can be represented as single tokens or comma-separated lists of tokens to indicate disjunction, e.g. `feature-type road` indicates only road features, while `feature-type road, fence` indicates roads *or* fences.

4 Sample Scenario

A sample scenario involves road creation with either the road tracker or ziplock snakes. Ziplock snakes can extract thin features, and can be applied to road extraction in low resolution images. The following rule packet has been written for ziplock snakes:

```
algorithm: (ziplock-snake image feature
                        &optional pane)

interactivity:          semiautomatic
accuracy:               coarse

arguments:
      image             image
      feature           road, fence, shoreline

requirements:
      task              extract
      feature-type      road
      gsd               > 1.5
```

Note that vocabulary items with numerical values can appear with inequalities.

Using these two rule packets, the HUB can now be invoked on a particular image to perform the road extraction task. Based on the GSD, the HUB will present to the user a menu with a list of algorithms that can be applied. Since, in this case, the GSD constraint is mutually exclusive between the two algorithms, the HUB will select either the Road Tracker or the ziplock snake. Selected algorithms are presented to the user through a pop-up menu, along with any advice that may be appropriate (e.g., if image conditions are on the edge of the robust performance range for an algorithm, the user will be told of those conditions). The user may then use any of the selected algorithms to complete the task.

The user also can pick a task and an algorithm, and ask the HUB which images are appropriate.

It should be possible to submit rule packets for an algorithm via a forms interface on the World Wide Web. The submission of rule packets will allow the addition of comments, or extra constraints which cannot be constructed using the vocabularies described here. The use of Prolog allows easy insertion of new rules and predicates to account for unforseen constraints.

References

[1] Aloimonos, Y. Purposive and Qualitative Action Vision. In *Proc. DARPA Image Understanding Workshop*, pages 816–828, Pittsburgh, PA, September 1990.

[2] Clément V., G. Giraudon, S. Houzelle, and F. Sandakly. Interpretation of Remotely Sensed Images in a Context of Multisensor Fusion using a Multispecialist Architecture. *IEEE Transactions on Geoscience and Remote Sensing*, 31(4), July 1989.

[3] Fu, D. D., K. J. Hammond, and M. J. Swain. Vision and Navigation in Man-Made Environments: Looking for Syrup in all the Right Places. In *Proceedings of the Workshop on Visual Behaviors*, pages 20–26, Seattle, WA, June 1994.

[4] Fua, P. Parametric Models are Versatile: The Case of Model Based Optimization. In *ISPRS WG III/2 Joint Workshop*, Stockholm, Sweden, September 1995.

[5] Fua, P. and C. Brechbuhler. Imposing Hard Constraints on Soft Snakes. In *European Conference on Computer Vision*, pages 495–506, Cambridge, England, April 1996. Available as Tech Note 553, Artificial Intelligence Center, SRI International.

[6] Fua, P. and Y. G. Leclerc. Model Driven Edge Detection. *Machine Vision and Applications*, 3:45–56, 1990.

[7] Fua, P. and Y. G. Leclerc. Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading. *International Journal of Computer Vision*, 16:35–56, September 1995.

[8] Fua, P. and Y. G. Leclerc. Taking Advantage of Image-Based and Geometry-Based Constraints to Recover 3–D Surfaces. *Computer Vision and Image Understanding*, 64(1):111–127, July 1996. Also available as Tech Note 536, Artificial Intelligence Center, SRI International.

[9] Fua, P. and T. Strat. Model-Based and Context-Based Vision at SRI. In *DARPA Image Understanding Workshop*, Palm Springs, CA, February 1996. Morgan Kaufmann.

[10] Gerson, D. RADIUS: The Government Viewpoint. In *ARPA Image Understanding Workshop*, Jan. 1992.

[11] Jain, R. C. and T. O. Binford. Ignorance, Myopia, and Naiveté in Computer Vision Systems. *CVGIP: Image Understanding*, 53(1):112–117, Jan. 1991.

[12] Kass, M., A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[13] Mundy, J. and P. Vrobel. The Role of IU Technology in RADIUS Phase II. In *Unpublished*, May 1994.

[14] Mundy, J., R. Welty, L. Quam, T. Strat, W. Bremmer, M. Horwedel, D. Hackett, and A. Hoogs. The RADIUS Common Development Environment. In *DARPA Image Understanding Workshop*, pages 215–226, San Diego,CA, 1992. Morgan Kaufmann.

[15] Neuenschwander, W., P. Fua, G. Székely, and O. Kubler. Making Snakes Converge from Minimal Initialization. In *DARPA Image Understanding Workshop*, Monterey, CA, November 1994. Morgan Kaufmann.

[16] Quam, L. H. Road Tracking and Anomaly Detection. In *DARPA Image Understanding Workshop*, pages 51–55. Morgan Kaufmann, May 1978.

[17] Strat, T. M. and W. D. Climenson. RADIUS: Site Model Content. In *ARPA Workshop on Image Understanding*, Nov. 1994.

[18] Strat, T. M. and M. A. Fischler. Context-Based Vision: Recognizing Objects Using Both 2D and 3D Imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1050–1065, Oct. 1991.

[19] Strat, T. M. and M. A. Fischler. The Role of Context in Computer Vision. In *ICCV Workshop on Context-Based Vision*, Cambridge, MA, June 1995.

[20] Terzopoulos, D., A. Witkin, and M. Kass. Symmetry-Seeking Models and 3D Object Reconstruction. *International Journal of Computer Vision*, 1:211–221, 1987.